

Project Documentation

FINAL IOT PROJECT

MIHAI COSTIN SERBAN

Contents

Project Overview and Executive Summary	2
Project Scopes and Objectives	2
Project Scopes	2
Project Objectives.....	3
Deliverables.....	4
Phase 1: IoT Dashboard for LED Control	4
Phase 2: IoT Dashboard for Temperature and Humidity Control.....	4
Phase 3: Light Intensity Capture and LED Activation	5
Phase 4 Task #1: User Profiles, RFID Tags, and Bluetooth Device Detection	6
Phase 4 Task #2: Nearby Bluetooth Devices Detection.....	6
Requirements.....	7
Phase 1:	7
Phase 2:	7
Phase 3:	7
Phase 4:	7
Work Breakdown Structure.....	8
Teamwork breakdown	8
Project Block Diagram	9
Method and Solution	9
Step 1: Library installation	9
Step 2: Hardware installation	12

Project Overview and Executive Summary

The general public can access existing technologies more readily as they develop, and as new ones continue to emerge. People use a wide range of technological gadgets in their daily lives, regardless of their past technical or programming expertise. Technology integration in residential environments, such as "smart homes," refers to the combination of digital and real-world components via an Internet of Things (IoT) framework. In this context, "smart home" refers to a system that establishes a more convenient and efficient living environment through the interconnection of numerous gadgets. This project is meant to be a prototype for creating a smart home. We use a Raspberry Pi, an online dashboard, and a variety of IoT devices to achieve this goal.

Project Scopes and Objectives

Project Scopes

Home Automation: The project aims to automate various aspects of a home, including controlling the fan, monitoring temperature and humidity, managing light intensity, and integrating RFID-based user identification.

Sensor Integration: The system incorporates sensors such as DHT11 for temperature and humidity monitoring, a light sensor for measuring light intensity, and an RFID sensor for user identification.

Actuator Control: The project controls actuators like a fan motor and an LED light based on sensor readings and user-defined thresholds.

User Interface: The web-based user interface provides real-time information about temperature, humidity, light intensity, fan status, and Bluetooth device count. It also displays user information, including a profile picture.

Email Notification: The system sends email notifications for certain events, such as the temperature exceeding a threshold and the light being turned on.

MQTT Communication: MQTT (Message Queuing Telemetry Transport) is used for communication between the Raspberry Pi and other devices, such as ESP8266 microcontrollers.

Database Integration: The project integrates with a MySQL database to store and retrieve user information, including temperature and light intensity thresholds.

Bluetooth Device Count: The system scans for Bluetooth devices and displays the count on the user interface.

Project Objectives

Temperature and Humidity Monitoring: Continuously monitor and display temperature and humidity levels inside the home.

Light Intensity Measurement: Measure and display the light intensity in the environment.

Fan Control: Automatically controls the fan based on temperature thresholds.

User Identification: Use RFID technology for user identification when someone enters the home.

Email Notifications: Send email notifications for critical events, such as high temperature and LED status changes.

User Profile Management: Display user information, including a profile picture, on the web interface.

Bluetooth Device Count: Scan for and display the count of Bluetooth devices in the vicinity.

MQTT Communication: Establish communication between the Raspberry Pi and ESP8266 devices using the MQTT protocol.

Database Interaction: Retrieve user information from the database based on RFID identification.

User Interface Design: Create an intuitive and responsive web-based user interface for monitoring and controlling the smart home system.

Deliverables

Phase 1: IoT Dashboard for LED Control

Create an IoT dashboard that can turn on and off a led on the breadboard with a simple click.

Data Capture:

Utilize a switch button on the Dashboard to capture data.

The switch has two states, ON and OFF, which represent the LED's status.

Data Communication:

Transmit captured data (Switch State) to a Raspberry Pi (RPi).

Data Presentation:

Develop a Dashboard that displays captured data, including the switch state and LED status.

Enable users to toggle the LED status from ON to OFF and vice versa.

Incorporate a light icon on the webpage to visually indicate the LED status.

Phase 2: IoT Dashboard for Temperature and Humidity Control

Create an IoT dashboard displaying temperature and humidity using gauges. When the temperature exceeds a certain threshold, an automatic email is sent to the user, asking if they want to start the fan. If the user agrees, the system activates the fan and updates its status on the dashboard. If not, no action is taken.

Data Capture:

Employ a DHT-11 sensor to capture current temperature and humidity.

Data Communication:

Transfer the captured data to an RPi for processing.

Data Presentation:

Design an IoT Dashboard showcasing temperature and humidity with gauges.

Implement email functionality: If temperature exceeds 24 degrees, send an email to the user.

If the user replies with YES, activate the fan and update its status on the dashboard.

Display fan status on the dashboard accordingly.

Phase 3: Light Intensity Capture and LED Activation

In this phase, you need to use a photoresistor with the ESP8266/ESP32 to measure light.

We then sent this data to the Raspberry Pi and MQTT broker. If the light went below 400, we turned on an LED and sent an email. The dashboard had to show the current light level, its status, and confirm the email was sent.

Data Capture:

Utilize a photoresistor sensor to capture light intensity connected to ESP8266 board.

Data Communication:

Transfer captured light intensity data to Raspberry Pi via Wi-Fi and MQTT broker.

Data Presentation:

Display captured light intensity, LED status, and email status on the dashboard.

If the light intensity drops below 400, turn on the LED and send an email.

Phase 4 Task #1: User Profiles, RFID Tags, and Bluetooth Device Detection

Task #1 involves creating user profiles with RFID tag numbers, temperature thresholds, and light intensity thresholds. When an RFID tag is read, the system updates thresholds based on the user's profile and sends an email notifying the entry time. The dashboard displays user profile information.

Data Capture:

Establish user profiles with RFID tag numbers, temperature thresholds, and light intensity thresholds in a database table.

Data Communication:

Transfer RFID tag information via MQTT broker.

Data Presentation:

Create a Dashboard box displaying user profile information.

Update thresholds and send an email notification when an RFID tag is read.

Phase 4 Task #2: Nearby Bluetooth Devices Detection

Task #2 of phase 4 focuses on counting nearby Bluetooth-enabled devices.

Data Capture:

Identify Bluetooth addresses of nearby devices using Raspberry Pi's built-in Bluetooth card.

Data Communication:

Retrieve data from Raspberry Pi for processing.

Data Presentation:

Showcase the count of detected Bluetooth-enabled devices on the dashboard.

Requirements

Phase 1:

Hardware: LED, Resistor, Wires, Breadboard Raspberry-Pi, GPIO Extension Board & Ribbon Cable

Software: Dash Plotly & Dash Libraries, Python and Python Text Editor, Raspberry Pi Libraries (RPI.GPIO, time)

Phase 2:

Hardware: DHT 11 Temperature and Humidity sensor, Resistors, Wires, Breadboard, Raspberry-Pi, DC Motor, L293D motor driver IC, Power Supply/Battery

Software: Dash Plotly & Dash Libraries, Freenove python code for DHT11, Python & Python Text Editor, Gmail account, Send/Receive with Python Libraries (smtplib, ssl, getpass, imaplib, email), Raspberry Pi Libraries (RPI.GPIO, time)

Phase 3:

Hardware Photoresistor, Resistors, Wires, Breadboard, Raspberry-Pi, ESP8266 with cable and a LED

Software requirements: MQTT and MQTT Libraries (paho.mqtt, PubSubClient), Python and Python IDE, Dash Plotly & Dash Libraries, Raspberry Pi Libraries (RPI.GPIO, time), C and Arduino IDE, Outlook, Send/Receive with Python Libraries (smtplib, ssl, getpass, imaplib, email), ESP8266Wifi Library

Phase 4:

Hardware: Resistors, Wires, Breadboard, Raspberry-Pi, ESP8266 with cable, Rfid Reader

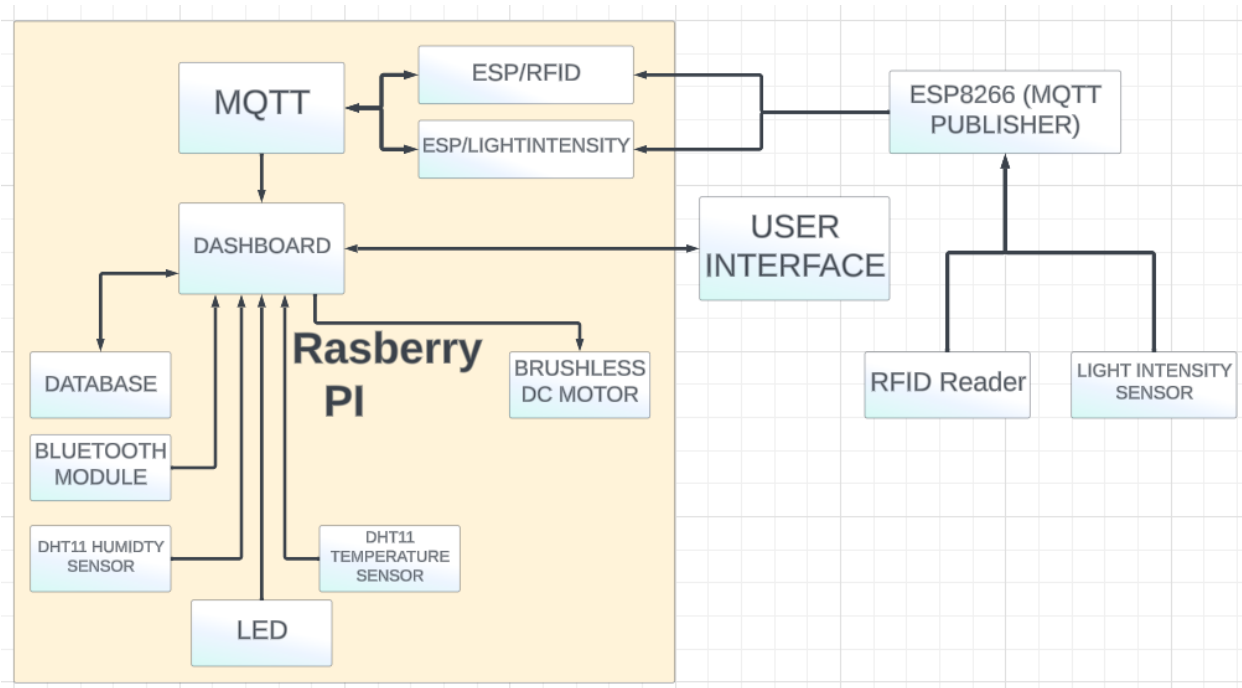
Software: C and Arduino IDE, Python and Python IDE, Raspberry Pi Libraries (RPI.GPIO, time), ESP8266Wifi Library, Bluetooth (bluetoothctl & bluez), MQTT and MQTT Libraries (paho.mqtt, PubSubClient), Dash Plotly & Dash Libraries, Outlook, Send/Receive with Python Libraries (smtplib, ssl, getpass, imaplib, email), and MariaDB.

Work Breakdown Structure

Teamwork breakdown

Phase1	Phase2
Mihai	Mihai
Initial Setup and basic dashboard	Debugging and Wiring issues fixed
Jerico	Jerico
Found Picture and click.	Designed the wiring.
Sayem	Sayem
Overview and debug	Added code to dashboard.py.
Phase 3	Phase 4
Jerico and Sayem did most of the work	Mihai did everything
Mihai only helped debugging in class.	Documentation
	PowerPoint
	Dashboard and RFID addition
	GitHub setup
	Fritzing wiring
	Etc.

Project Block Diagram



Method and Solution

To recreate the project, you will not have to recreate every single phase of the project and then combine them as we have already done so. You will only need to follow the steps provided:

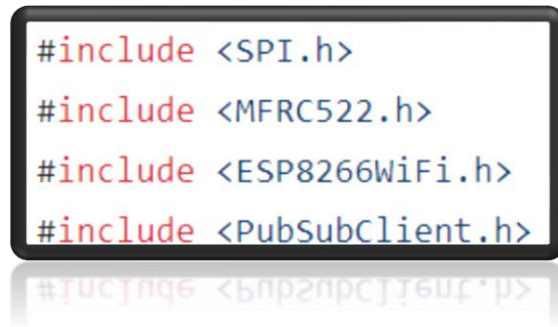
All code can be found here: https://github.com/laronichill/iot_project

Step 1: Library installation

- First install python3, which should already be installed on Raspberry Pi from factory
Can be found here: <https://www.python.org/downloads/>
- Next, we need to install **pip3**. On Linux/raspberry pi, run this command:
`$sudo apt install python3-pip`
- To install Dash and all its libraries for the user interface
 - Dash itself
`$ pip3 install dash`
<https://pypi.org/project/dash/>

- Collection of Plotly figure templates customized for Bootstrap themes:
\$ pip3 install dash-bootstrap-templates
<https://pypi.org/project/dash-bootstrap-templates/>
- Bootstrap components for Plotly Dash
\$ pip3 install dash-bootstrap-components
<https://pypi.org/project/dash-bootstrap-components/>
- For Gifs
\$ pip3 install dash-extensions
<https://pypi.org/project/dash-extensions/>
- DAQ components for Dash
\$ pip3 install dash-daq
<https://pypi.org/project/dash-daq/>
- Next, create a Gmail account with an API app password.
- This is optional, but you can download the Freenove_DHT python script for the temperature and humidity DHT11 sensor, but it can already be found in the code repository.
- Next have need PyMySQL
\$ python3 -m pip install PyMySQL
- Bluetooth libraries
\$ sudo apt install bluetooth
\$ sudo apt install bluez
- Install the latest Arduino IDE to run code on the ESP8266
\$ sudo apt install Arduino

- You also need to install the following libraries on the Arduino IDE using the library manager:



```
#include <SPI.h>
#include <MFRC522.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
```

SPI:

The SPI library is already included in the Arduino IDE.

MFRC522:

Sketch > Include Library > Manage Libraries, search for **MFRC522**

More information: <https://www.arduino.cc/reference/en/libraries/mfrc522/>

ESP8266Wifi:

Click Sketch > Include Library > Manage Libraries, search for **ESP8266WiFi** and then install with the latest version.

PubSubClient:

Click Sketch > Include Library > Manage Libraries, search for **PubSubClient** and then install with the latest version.

- Next we need to install Mosquitto and run basic configurations. Open terminal and run these commands.
 - First update system
\$ sudo apt update
 - Install Mosquitto Broker
\$ sudo apt install -y mosquitto mosquitto-clients

- To make Mosquitto auto-start on boot

\$ sudo systemctl enable mosquitto.service

- Test installation

\$ mosquitto -v

- To have remote access:

\$ sudo nano /etc/mosquitto/mosquitto.conf

Now append the file with these two lines.

listener 1883 0.0.0.0

allow_anonymous true

Exit nano with Ctrl+X and save edit

- Restart the mosquitto server to add config to runtime

\$ sudo systemctl restart mosquitto

More information: <https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/>

- Now install Paho MQTT with this terminal command on your Raspberry Pi

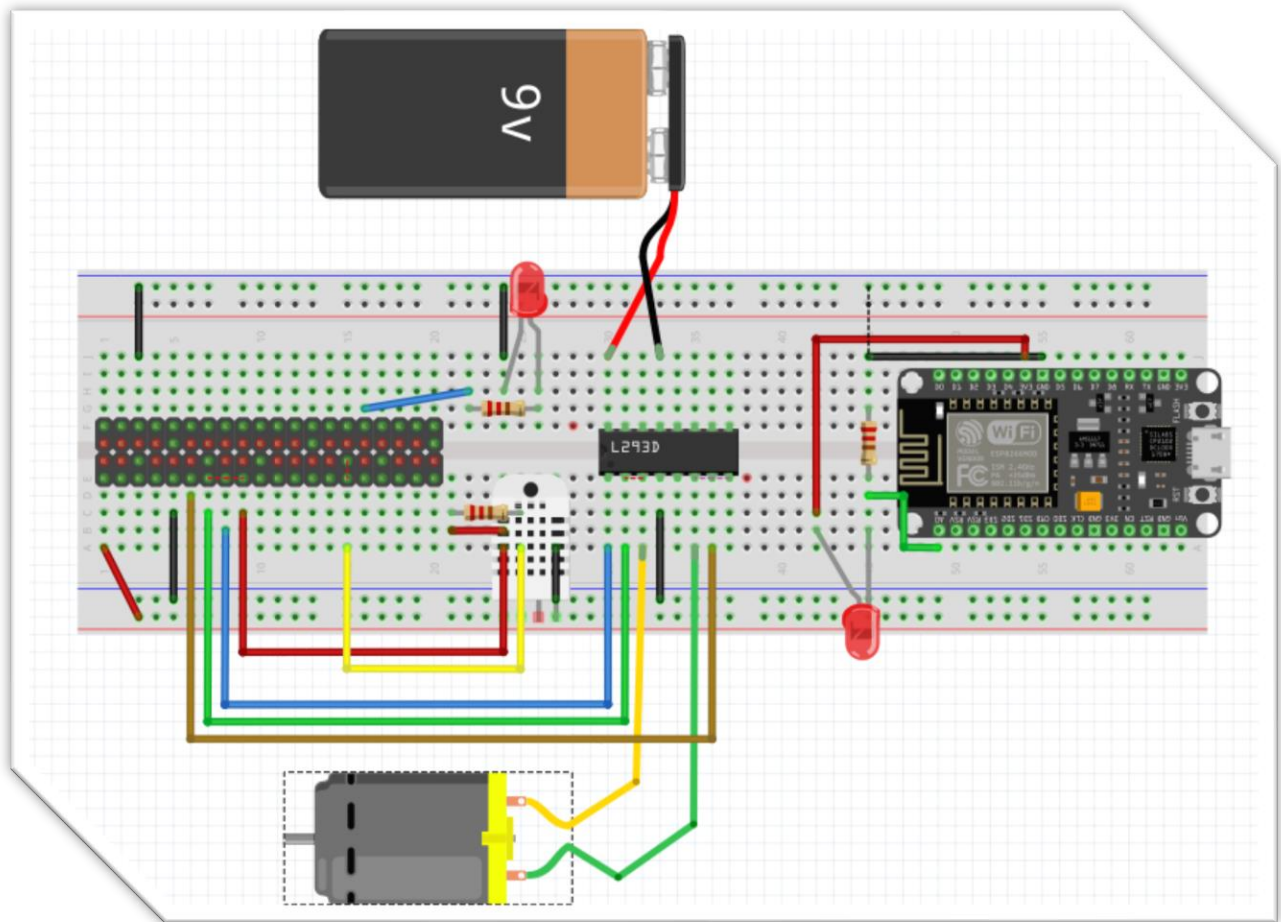
\$ pip install paho-mqtt

- Install MariaDB

\$ sudo apt install mariadb-server

Step 2: Hardware installation

Recreate the following circuit on your breadboards:



Missing connections:

- Connect ESP8266 to Raspberry Pi using a Micro-USB cable
- Connect GPIO T-Extension to your Raspberry-Pi

Step 3: uploading and running Codes.

- Before running anything, we need to setup the database. Open the `database_creation_and_reference_script.py` and follow the guide there.
- Open the `RFID\RFID.ino` file with Arduino IDE.
 - Edit the ssid, password with that of your wifi.
 - Edit the mqtt_server ip with that of your current raspberry pi machine. To figure out what it is, simply run the `$ ifconfig` in terminal.
 - Now upload the file to the ESP8266 Node Mcu.
- Now run the `Dashboard.py` file inside the repository using any method you like such as the terminal `$ python3 Dashboard.py` or open the file with an ide and run from there. Best IDE to use is Thonny.

Step 4: Last Step

Simply enjoy the Smart Home Dashboard and all its functionalities.

Link to GitHub repo:

https://github.com/laronichill/iot_project